|  |  |  |
| --- | --- | --- |
| WHStore:watsonhall:groups:owasp:OWASP_Image_Toolbox:owasp_logo_122106.eps  **Cornucopia**  **Ecommerce Website Edition v0.3**  OWASP Cornucopia is a mechanism to assist software development teams identify security requirements in Agile, conventional and formal development processes  Author  Colin Watson  Acknowledgments  Microsoft SDL Team for the Elevation of Privilege Threat Modelling Game, published under a Creative Commons Attribution license, as the inspiration for Cornucopia and from which many ideas, especially the game theory, were copied.  Keith Turpin and contributors to the “OWASP Secure Coding Practices - Quick Reference Guide”, originally donated to OWASP by Boeing, which is used as the primary source of security requirements information to formulate the content of the cards.  Contributors, supporters, sponsors and volunteers to the OWASP ASVS and AppSensor projects, the Common Attack Pattern Enumeration and Classification (CAPEC), and SAFECode’s “Practical Security Stories and Security Tasks for Agile Development Environments” which are all used in the cross-references provided.  Playgen for providing an illuminating afternoon seminar on task gamification, and tartanmaker.com for the online tool to help create the card back pattern.  OWASP does not endorse or recommend commercial products or services  © 2012 OWASP Foundation  This document is licensed under the Creative Commons Attribution-ShareAlike 3.0 license | |  |
| **Introduction**  The idea behind Cornucopia was to help development teams, especially those using Agile methodologies, to identify application security requirements and develop security-based user stories. Although the idea had been waiting for enough time to progress it, the final motivation came when [SAFECode](http://www.safecode.org/) published its [Practical Security Stories and Security Tasks for Agile Development Environments](http://www.safecode.org/publications/SAFECode_Agile_Dev_Security0712.pdf) in July 2012.  The Microsoft SDL team had already published its super [Elevation of Privilege: The Threat Modeling Game](http://www.microsoft.com/security/sdl/adopt/eop.aspx) (EoP) but that did not seem to address the right sort of level of issues that web application development teams mostly have to address. It was a great concept and game strategy, and thankfully it was [published under a](http://blogs.msdn.com/b/sdl/archive/2010/03/02/announcing-elevation-of-privilege-the-threat-modeling-game.aspx) [Creative Commons Attribution License](http://creativecommons.org/licenses/by/3.0/).  Cornucopia Ecommerce Website Edition is based the concepts and game ideas in EoP, but has been modified to be more relevant to the types of issues ecommerce website developers encounter. It also attempts to introduce threat-modelling ideas into development teams which are not familiar with STRIDE and DREAD, or use Agile methodologies, or are more focused on web application weaknesses than other types of software vulnerabilities.  **The deck**  Instead of EoP’s STRIDE suits, the suits were selected based on the structure of the [OWASP Secure Coding Practices - Quick Reference Guide](https://www.owasp.org/index.php/OWASP_Secure_Coding_Practices_-_Quick_Reference_Guide) (SCP), but with additional consideration of the sections in the [OWASP Application Security Verification Standard](https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project), the [OWASP Testing Guide](https://www.owasp.org/index.php/OWASP_Testing_Project) and David Rook’s [Principles of Secure Development](http://www.securityninja.co.uk/secure-development/the-principles-place/). These provided five suits, and a sixth called “Cornucopia” was created for everything else:   * Data validation and encoding * Authentication * Session management * Authorization * Cryptography * Cornucopia   Each suit contains 13 cards (Ace, 2-10, Jack, Queen and King) but, unlike EoP, there are also two Joker cards. The content was mainly drawn from the SCP.  **Mappings**  The other driver for Cornucopia was to try to link the attacks with requirements and verification techniques. An initial aim had been to reference [CWE](http://cwe.mitre.org/) weakness IDs, but these proved too numerous, and instead it was decided to map each card to [CAPEC](http://capec.mitre.org/) software attack pattern IDs which are already mapped to CWEs so the desired result is achieved.  Each card is also mapped to the 36 primary security stories in the SAFECode’s document, as well as to the OWASP SCP v2, ASVS 2009 and [AppSensor](https://www.owasp.org/index.php/OWASP_AppSensor_Project) (application attack detection and response). Combined, I hope these will help teams create their own security-related stories that can be consumed in Agile processes. |  | **Game strategy**  Apart from the content differences, the game rules are virtually identical to [those for EoP](http://social.technet.microsoft.com/wiki/contents/articles/285.elevation-of-privilege-the-game.aspx).  **Printing the cards**  The cards can be printed in black & white but are more effective in color. The cards in the later pages of this document have been laid out to fit on one type of pre-scored business card sheets. This appeared to be the quickest way to provide to create playing cards quickly. Avery product code C32030 has been tested successfully, but any 10 up 85mm x 54 mm cards on A4 paper should work with a little adjustment. Other stationery suppliers like Ryman and Sigel produce similar sheets. These card sheets are not inexpensive, so care should be taken in deciding what to print and using what media and printer type.  The cards can of course just be printed on any paper or card and then cut-up manually, or a commercial printer would be able to print larger volumes and cut the cards to size. The cut lines are shown on the penultimate page of this document, but Avery also produce a landscape A4 template ([A-0017-01\_L.doc](http://www.avery.co.uk/avery/secure/gb_softwaredownload?downloadPath=%2Fuk%2FA-0017-01_L.doc)) that can be used as a guide.  An optional card back design (in OWASP tartan) has been provided as the last page of this document. There is no special alignment needed. Dual-sided printing needs special care taken.  You could customize the cards or the backs for your own organization’s preferences.  **Customization**  After you have used Cornucopia a few times, you may feel that some cards are less relevant to your applications, or the threats are different for your organization. Edit this document yourself to make the cards more suitable for your teams, or create new decks completely.  **Provide feedback**  If you have ideas or feedback on the use of OWASP Cornucopia, please share them. Even better if you create alternative versions of the cards, or produce professional print-ready versions, please share that with the volunteers who created this edition and with the wider application development and application security community.  The best place to use to discuss or contribute is the mailing list for the OWASP project “Secure Coding Practices - Quick Reference Guide”   * Mailing list <https://lists.owasp.org/mailman/listinfo/owasp-secure-coding-practices> * Project home page <https://www.owasp.org/index.php/OWASP_Secure_Coding_Practices_-_Quick_Reference_Guide>   All OWASP documents and tools are free to download and use. OWASP Cornucopia is licensed under the Creative Commons Attribution-ShareAlike 3.0 license. |

|  |  |  |
| --- | --- | --- |
| **Instructions**  The text on each card describes an attack, but the attacker is given a name, which are unique across all the cards. The name can represent a computer system (e.g. the database, the file system, another application, a related service, a botnet), an individual person (e.g. a citizen, a customer, a client, an employee, a criminal, a spy), or even a group of people (e.g. a competitive organization, activists with a common cause). The attacker might be remote in some other device/location, or local/internal with access to the same device, host or network as the application is running on. The attacker is always named at the start of each description. An example is:  *William has control over the generation of session identifiers*  This means the attacker (William) can create new session identifiers that the application accepts.  The attacks were primarily drawn from the security requirements listed in the SCP????, v1.1 but then supplemented with verification objectives from the OWASP “Application Security Verification Standard for Web Applications (2009)”, the security focused stories in SAFECode’s “Practical Security Stories and Security Tasks for Agile Development Environments”, and finally a review of the cards in EOP.  Lookups between the attacks and five resources are provided on most cards:  A look-up means the attack is included within the referenced item, but does not necessarily encompass the whole of its intent. For structured data like CAPEC, the most specific reference is provided but sometimes a cross-reference is provided that also has more specific (child) examples. There are no lookups on the six Aces and two Jokers. Instead these cards have some general tips in italic text.  **Preparations**   1. Print out a deck of Cornucopia cards (see previous page) 2. Identify an application or application process to review; this might be a concept, design or an actual implementation 3. Create a data flow diagram 4. Identify and invite a group of 3-8 architects, developers, testers and other business stakeholders together and sit around a table 5. Have some prizes to hand (gold stars, chocolate, beer or flowers depending upon your office culture) |  | **Play**  One suit - *Cornucopia* - acts as trumps. Aces are high (i.e. they beat Kings). It helps if there is someone dedicated to documenting the results.   1. Remove the Jokers and a few low-score (2, 3, 4) cards from *Cornucopia* suit to ensure each player will have the same number of cards 2. Shuffle the pack and deal all the cards 3. The person with the card “2 of *Data Validation and Encoding*” starts the first round, with the lead suit being *Data Validation and Encoding* 4. To play a card, you read it out aloud, record it on the score card, and explain how (or not) the threat could apply (the player gets a point for attacks that work, and the group thinks it is an actionable bug) - don’t try to think of mitigations at this stage, and don’t exclude a threat just because it is believed it is already mitigated 5. Play clockwise, each person must play a card in the same way; if you have any card of the matching lead suit you must play one of those, otherwise they can play a card from any other suit. Only a higher card of the same suit, or the highest card in the trump suit *Cornucopia*, wins the hand. 6. The person who wins the round, leads the next round (i.e. they play first), and thus defines the next lead suit 7. Repeat until all the cards are played   **Scoring**  The objective is to identify applicable threats, and win hands (rounds):   1. Score +1 for each card you can identify as a valid threat to the application under consideration 2. Score +1 if you win a round 3. Once all cards have been played, whoever has the most points wins   **Closure**   1. Review all the applicable threats and the matching security requirements 2. Create user stories, specifications and test cases as required for your development methodology   **Alternatives**  If you are new to the game, remove the two Joker cards to begin with. Add the Joker cards back in once people become more familiar with the process.  Practice on an imaginary application, or even a future planned application, rather than trying to find fault with existing applications until the participants are happy with the usefulness of the game.  Consider just playing with one suit to make a shorter session – but try to cover all the suits for every project. |

**Score card 1/3 - Requirements**

| No | Card | Player | Notes on Requirement |  | No | Card | Player | Notes on Requirement |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 |  |  |  |  | 21 |  |  |  |
| 2 |  |  |  |  | 22 |  |  |  |
| 3 |  |  |  |  | 23 |  |  |  |
| 4 |  |  |  |  | 24 |  |  |  |
| 5 |  |  |  |  | 25 |  |  |  |
| 6 |  |  |  |  | 26 |  |  |  |
| 7 |  |  |  |  | 27 |  |  |  |
| 8 |  |  |  |  | 28 |  |  |  |
| 9 |  |  |  |  | 29 |  |  |  |
| 10 |  |  |  |  | 30 |  |  |  |
| 11 |  |  |  |  | 31 |  |  |  |
| 12 |  |  |  |  | 32 |  |  |  |
| 13 |  |  |  |  | 33 |  |  |  |
| 14 |  |  |  |  | 34 |  |  |  |
| 15 |  |  |  |  | 35 |  |  |  |
| 16 |  |  |  |  | 36 |  |  |  |
| 17 |  |  |  |  | 37 |  |  |  |
| 18 |  |  |  |  | 38 |  |  |  |
| 19 |  |  |  |  | 39 |  |  |  |
| 20 |  |  |  |  | 40 |  |  |  |

**Score card 2/3 - Requirements**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No | Card | Player | Notes on Requirement |  | No | Card | Player | Notes on Requirement |
| 41 |  |  |  |  | 61 |  |  |  |
| 42 |  |  |  |  | 62 |  |  |  |
| 43 |  |  |  |  | 63 |  |  |  |
| 44 |  |  |  |  | 64 |  |  |  |
| 45 |  |  |  |  | 65 |  |  |  |
| 46 |  |  |  |  | 66 |  |  |  |
| 47 |  |  |  |  | 67 |  |  |  |
| 48 |  |  |  |  | 68 |  |  |  |
| 49 |  |  |  |  | 69 |  |  |  |
| 50 |  |  |  |  | 70 |  |  |  |
| 51 |  |  |  |  | 71 |  |  |  |
| 52 |  |  |  |  | 72 |  |  |  |
| 53 |  |  |  |  | 73 |  |  |  |
| 54 |  |  |  |  | 74 |  |  |  |
| 55 |  |  |  |  | 75 |  |  |  |
| 56 |  |  |  |  | 76 |  |  |  |
| 57 |  |  |  |  | 77 |  |  |  |
| 58 |  |  |  |  | 78 |  |  |  |
| 59 |  |  |  |  | 79 |  |  |  |
| 60 |  |  |  |  | 80 |  |  |  |

**Score card 3/3 - Players**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Requirements | | Rounds | | Total | Rank |
|  | Tally | Sub-total | Tally | Sub-total |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Data Validation & Encoding** | **A** | **Data Validation & Encoding** |  | **Data Validation & Encoding** | **2** | **Data Validation & Encoding** | **3** |
| You have invented a new attack against Data Validation and Encoding | (no card) | Brian can gather information about the underlying configurations, schemas, logic, code, software, services and infrastructure due to the content of error messages, or due to poor configuration, or due to the presence of default installation files or old, test, backup or copies of resources, or exposure of source code | Robert can input malicious structured or unstructured data because the allowed protocol format is not being checked, or the structure is not being verified, or the individual data elements are not being validated for format, type, range, length and a whitelist of allowed characters or formats |
| *Read more about this topic in OWASP’s free Cheat Sheets on Input Validation, XSS (Cross Site Scripting) Prevention, DOM-based XSS Prevention, SQL Injection Prevention, and Query Parameterization* |  | |  | | --- | | OWASP SCP  69, 107-109, 136, 137, 153, 156, 158, 162 | | OWASP ASVS  4.5, 8.1, 8.2 | | OWASP AppSensor  HT1-3 | | CAPEC  54, 224 | | SAFECode  4, 23 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  8, 9, 11-14, 16, 159, 190, 191 | | OWASP ASVS  5.2 | | OWASP AppSensor  RE7-8, AE4-7, IE2-3,CIE1,CIE3-4,HT1-3 | | CAPEC  28,48,126,165,213,220,221,257,261,271,272 | | SAFECode  3, 16, 24, 35 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | |
| **Data Validation & Encoding** | **4** | **Data Validation & Encoding** | **5** | **Data Validation & Encoding** | **6** | **Data Validation & Encoding** | **7** |
| Dave can input malicious data because it is not being checked within the context of the current user and process | Jee can bypass the centralized encoding routines since they are not being used comprehensively, or the wrong encodings are being used for the context | Jason can bypass the centralized validation routines since they are not being used comprehensively on all inputs | Jan can craft special payloads to foil input validation because the character set is not specified/enforced, or the data is encoded multiple times, or the data is not fully converted into the same format the application uses (e.g. canonicalization) before being validated, or variables are not strongly typed |
| |  | | --- | | OWASP SCP  8, 10, 183 | | OWASP ASVS  5.2, 11.1 | | OWASP AppSensor  RE3-6,AE8-11,SE1,3-6,IE2-4,HT1-3 | | CAPEC  28, 31, 48, 126, 162, 165, 213, 220, 221,261 | | SAFECode  24, 35 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  3, 15, 18, 19, 168 | | OWASP ASVS  6.9 | | OWASP AppSensor  - | | CAPEC  28, 31, 152, 160, 468 | | SAFECode  2, 17 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  3, 168 | | OWASP ASVS  5.2, 5.6, 6.9 | | OWASP AppSensor  IE2-3 | | CAPEC  28 | | SAFECode  3, 16, 24 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  4, 5, 7, 150 | | OWASP ASVS  5.4, 5.8, 10.9 | | OWASP AppSensor  IE2-3, EE1-2 | | CAPEC  28, 153, 165 | | SAFECode  3, 16, 24 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Data Validation & Encoding** | **8** | **Data Validation & Encoding** | **9** | **Data Validation & Encoding** | **10** | **Data Validation & Encoding** | **J** |
| Sarah can bypass the centralized sanitization routines since they are not being used comprehensively | Shamun can bypass input validation or output validation checks because validation failures are not rejected or sanitized | Jerry can exploit the trust the application places in a source of data (e.g. user-definable data, manipulation of locally stored data, alteration to state data on a client device, lack of verification of identity such as Jerry can pretend to be Colin) | Dennis has control over input validation, output validation or output encoding code/routines so they can be bypassed |
| |  | | --- | | OWASP SCP  15, 169 | | OWASP ASVS  6.9, 8.7 | | OWASP AppSensor  - | | CAPEC  28, 31, 152, 160, 468 | | SAFECode  2, 17 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  6, 168 | | OWASP ASVS  5.3 | | OWASP AppSensor  IE2-3 | | CAPEC  28 | | SAFECode  3, 16, 24 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  2, 19, 92, 95, 180 | | OWASP ASVS  10.6 | | OWASP AppSensor  IE4, IE5 | | CAPEC  12, 51, 57, 90,111,145,194,195,202,218,463 | | SAFECode  14 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  1, 17 | | OWASP ASVS  5.5, 6.2 | | OWASP AppSensor  RE3, RE4 | | CAPEC  56, 87, 207 | | SAFECode  2, 17 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | |
| **Data Validation & Encoding** | **Q** | **Data Validation & Encoding** | **K** |  |  |  |  |
| Geoff can inject data into a client or device interpreter because a parameterised interface is not being used, or has not been implemented correctly, or the data has not been encoded correctly for the context, or there is no restrictive policy on code or data includes | Gabe can inject data into an server-side interpreter (e.g. SQL, OS commands, Xpath, Server JavaScript, SMTP) because a strongly typed parameterised interface is not being used or has not been implemented correctly | (no card) | (no card) |
| |  | | --- | | OWASP SCP  10, 15, 16, 19, 20 | | OWASP ASVS  6.1, 6.3, 6.8 | | OWASP AppSensor  IE1, RP3 | | CAPEC  28, 31, 152, 160, 468 | | SAFECode  2, 17 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  15, 19-22, 167, 180, 203, 210, 211 | | OWASP ASVS  6.3, 6.4, 6.5, 6.6, 6.7, 6.8 | | OWASP AppSensor  CIE1-2 | | CAPEC  23, 28, 76, 152, 160, 261 | | SAFECode  2, 19, 20 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Authentication** | **A** | **Authentication** |  | **Authentication** | **2** | **Authentication** | **3** |
| You have invented a new attack against Authentication | (no card) | James can undertake authentication functions (e.g. attempt to log in, log in with stolen credentials, reset the password) without the real user ever being aware this has occurred | Muhammad can obtain a user's password or other secrets such as security questions, by observation during entry, or from a local cache, or in transit, or by reading it from some unprotected location, or because it is widely known, or because it never expires, or because the user cannot change her own password |
| *Read more about this topic in OWASP’s free*  *Authentication Cheat Sheet* |  | |  | | --- | | OWASP SCP  47, 52 | | OWASP ASVS  2.12 | | OWASP AppSensor  UT1 | | CAPEC  - | | SAFECode  28 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  36-7, 40, 43, 48, 51, 119, 139-40, 146 | | OWASP ASVS  2.2, 2.8, 2.10, 8.10, 9.1 | | OWASP AppSensor  - | | CAPEC  37 | | SAFECode  28 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | |
| **Authentication** | **4** | **Authentication** | **5** | **Authentication** | **6** | **Authentication** | **7** |
| Sebastien can easily identify user names or can enumerate them | Javier can use default, test or easily guessable credentials to authenticate, or can use an old account or an account not necessary for the application | Sven can reuse a temporary password because the user does not have to change it on first use, or it has too long or no expiry | Cecilia can use brute force and dictionary attacks against one or many accounts without limit, or these attacks are simplified due to insufficient complexity, length, expiration and re-use requirements for passwords |
| |  | | --- | | OWASP SCP  33, 53 | | OWASP ASVS  - | | OWASP AppSensor  AE1 | | CAPEC  383 | | SAFECode  28 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  54, 175, 178 | | OWASP ASVS  - | | OWASP AppSensor  AE12, HT3 | | CAPEC  70 | | SAFECode  28 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  37, 45, 46, 178 | | OWASP ASVS  - | | OWASP AppSensor  - | | CAPEC  50 | | SAFECode  28 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  33, 38, 39, 41, 50, 53 | | OWASP ASVS  2.3 | | OWASP AppSensor  AE2, AE3 | | CAPEC  2, 16 | | SAFECode  27 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Authentication** | **8** | **Authentication** | **9** | **Authentication** | **10** | **Authentication** | **J** |
| Kate can by bypass authentication because it does not fail secure (i.e. it defaults to allowing access) | Claudia can undertake more critical functions because authentication requirements are too weak, or there is no requirement to re-authenticate for these | Pravin can bypass authentication controls because a centralized standard, tested and approved authentication module/framework/service, separate to the resource being requested, is not being used | Mark can access resources or services because there is no authentication requirement, or it was assumed authentication would be undertaken by some other system, or was performed in some previous action |
| |  | | --- | | OWASP SCP  28 | | OWASP ASVS  2.5 | | OWASP AppSensor  - | | CAPEC  115 | | SAFECode  28 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  55, 56 | | OWASP ASVS  2.6, 2.9 | | OWASP AppSensor  - | | CAPEC  21 | | SAFECode  14, 28 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  25, 26,27 | | OWASP ASVS  2.11 | | OWASP AppSensor  - | | CAPEC  90, 115 | | SAFECode  14, 28 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  23, 32, 34 | | OWASP ASVS  2.1 | | OWASP AppSensor  - | | CAPEC  115 | | SAFECode  14, 28 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | |
| **Authentication** | **Q** | **Authentication** | **K** |  |  |  |  |
| Jaime can bypass authentication because it is not enforced comprehensively across all entry points, modules, functions, content and other data, or is not applied with equal rigor for all types of authentication functionality (e.g. register, password change, password change, log out, administration) | Olga can influence or alter authentication code/routines so they can be bypassed | (no card) | (no card) |
| |  | | --- | | OWASP SCP  23, 29, 42, 49 | | OWASP ASVS  2.1, 2.7 | | OWASP AppSensor  - | | CAPEC  36, 50, 115, 121, 179 | | SAFECode  14, 28 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  24 | | OWASP ASVS  2.4 | | OWASP AppSensor  - | | CAPEC  115, 207 | | SAFECode  14, 28 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Session Management** | **A** | **Session Management** |  | **Session Management** | **2** | **Session Management** | **3** |
| You have invented a new attack against Session Management | (no card) | William has control over the generation of session identifiers | Ryan can use a single account in parallel since concurrent sessions are allowed |
| *Read more about this topic in OWASP’s free Cheat Sheets on Session Management, and Cross Site Request Forgery (CSRF) Prevention* |  | |  | | --- | | OWASP SCP  59 | | OWASP ASVS  3.9 | | OWASP AppSensor  SE2 | | CAPEC  31, 60, 61 | | SAFECode  28 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  68 | | OWASP ASVS  - | | OWASP AppSensor  - | | CAPEC  - | | SAFECode  28 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | |
| **Session Management** | **4** | **Session Management** | **5** | **Session Management** | **6** | **Session Management** | **7** |
| Alison can set session identification cookies on another web application because the domain and path are not restricted sufficiently | John can predict or guess session identifiers because they are not changed when the user's role alters (e.g. pre and post authentication) and when switching between non-encrypted and encrypted communications, or are not sufficiently long and random, or are not changed periodically | Gary can take over a user's session because there is a long or no inactivity timeout, or a long or no overall session time limit, or the same session can be used from more than one device/location | Casey can utilize Adam's session after he has finished, because there is no log out function, or he cannot easily log out, or log out does not properly terminate the session |
| |  | | --- | | OWASP SCP  59, 61 | | OWASP ASVS  3.12 | | OWASP AppSensor  SE2 | | CAPEC  31, 61 | | SAFECode  28 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  66, 67, 71, 72 | | OWASP ASVS  3.6, 3.7, 3.8, 3.11 | | OWASP AppSensor  SE4-6 | | CAPEC  31 | | SAFECode  28 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  64, 65 | | OWASP ASVS  3.3, 3.10 | | OWASP AppSensor  SE5, SE6 | | CAPEC  21 | | SAFECode  28 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  62, 63 | | OWASP ASVS  3.2, 3.4, 3.8 | | OWASP AppSensor  - | | CAPEC  21 | | SAFECode  28 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Session Management** | **8** | **Session Management** | **9** | **Session Management** | **10** | **Session Management** | **J** |
| Matt can abuse long sessions because the application does not require periodic re-authentication to check if privileges have changed | Ivan can steal session identifiers because they are sent over insecure channels, or are logged, or are revealed in error messages, or are included in URLs, or are accessible un-necessarily by code which the attacker can influence or alter | Marce can forge requests because per-session, or per-request for more critical actions, strong random tokens or similar are not being used for actions that change state | Jeff can resend an identical interaction (e.g. HTTP request, signal, button press) and it is accepted, not rejected |
| |  | | --- | | OWASP SCP  96 | | OWASP ASVS  - | | OWASP AppSensor  - | | CAPEC  21 | | SAFECode  28 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  69, 75, 76, 119, 138 | | OWASP ASVS  3.5, 8.10, 11.4 | | OWASP AppSensor  SE4-6 | | CAPEC  31, 60 | | SAFECode  28 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  73, 74 | | OWASP ASVS  11.7 | | OWASP AppSensor  IE4 | | CAPEC  62, 111 | | SAFECode  18 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  - | | OWASP ASVS  - | | OWASP AppSensor  IE5 | | CAPEC  60 | | SAFECode  12, 14 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | |
| **Session Management** | **Q** | **Session Management** | **K** |  |  |  |  |
| Salim can bypass session management because it is not applied comprehensively and consistently across the application | Peter can bypass the session management controls because they have been self-built and/or are weak, instead of using a standard framework or approved tested module | (no card) | (no card) |
| |  | | --- | | OWASP SCP  58 | | OWASP ASVS  3.1 | | OWASP AppSensor  - | | CAPEC  21 | | SAFECode  14, 28 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  58, 60 | | OWASP ASVS  3.1 | | OWASP AppSensor  - | | CAPEC  21 | | SAFECode  14, 28 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Authorization** | **A** | **Authorization** |  | **Authorization** | **2** | **Authorization** | **3** |
| You have invented a new attack against Authorization | (no card) | Tim can influence where data is sent or forwarded to | Christian can access (read, write, update or delete) information, which they should not have permission to, through another mechanism that does have permission (e.g. search indexer, logger, reporting), or because it is cached, or other information leakage |
| *Read more about this topic in OWASP’s Development and Testing Guides* |  | |  | | --- | | OWASP SCP  44 | | OWASP ASVS  4.1, 4.2, 4.3, 4.4, 4.6 | | OWASP AppSensor  - | | CAPEC  153 | | SAFECode  8, 10, 11 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  51, 139, 140, 150 | | OWASP ASVS  4.1, 8.7, 9.1, 9.2, 9.3, 9.4, 9.5 | | OWASP AppSensor  - | | CAPEC  69, 213 | | SAFECode  8, 10, 11 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | |
| **Authorization** | **4** | **Authorization** | **5** | **Authorization** | **6** | **Authorization** | **7** |
| Kelly can bypass authorization controls because they do not fail securely (i.e. they default to allowing access) | Chad can access resources (including services, processes, AJAX, Flash, video, images, documents, temporary files, session data, system properties, configuration data, registry settings, logs) he should not be able to due to missing authorization, or due to excessive privileges (e.g. not using the principle of least privilege) | Eduardo can access data he does not have permission to, even though he has permission to the form/page/URL/entry point | Yuanjing can access application functions, objects, or properties he is not authorized to access |
| |  | | --- | | OWASP SCP  79, 80 | | OWASP ASVS  4.8 | | OWASP AppSensor  - | | CAPEC  122 | | SAFECode  8, 10, 11 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP 30,70,81,83-4,87-9, 99,117,131-2,142,154,170,179,190-2 | | OWASP ASVS  4.1, 4.3, 4.4, 4.6, 8.7, 10.7 | | OWASP AppSensor  ACE1-4, HT2 | | CAPEC  75, 87, 95, 126, 149, 155, 203, 213, 264-5 | | SAFECode  8, 10, 11, 13 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  81 | | OWASP ASVS  4.1, 4.2, 4.3, 4.4, 4.6 | | OWASP AppSensor  ACE1-4 | | CAPEC  122 | | SAFECode  8, 10, 11 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  81, 85, 86 | | OWASP ASVS  4.1, 4.2, 4.3, 4.4, 4.6 | | OWASP AppSensor  ACE1-4 | | CAPEC  122 | | SAFECode  8, 10, 11 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Authorization** | **8** | **Authorization** | **9** | **Authorization** | **10** | **Authorization** | **J** |
| Tom can bypass business rules by altering the usual process sequence or flow, or by undertaking the process in the incorrect order, or by manipulating date and time values used by the application, or by using valid features for unintended purposes, or by otherwise manipulating control data | Mike can misuse an application by using a valid feature too fast, or too frequently, or other way that is not intended, or consumes the application's resources, or causes race conditions, or over-utilizes a feature | Richard can bypass the centralized authorization controls since they are not being used comprehensively on all interactions | Dinis can access security configuration information, or access control lists |
| |  | | --- | | OWASP SCP  10, 32, 93, 94, 189 | | OWASP ASVS  4.1, 4.2, 4.3, 4.4, 4.6, 4.12 | | OWASP AppSensor  ACE3 | | CAPEC  25, 39, 74, 162, 166, 207 | | SAFECode  8, 10, 11, 12 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  94 | | OWASP ASVS  4.12 | | OWASP AppSensor  AE3, FIO1-2, UT2-4, STE1-3 | | CAPEC  26, 29, 119, 261 | | SAFECode  1, 35 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  78, 91 | | OWASP ASVS  4.13, 4.14 | | OWASP AppSensor  ACE1-4 | | CAPEC  36, 95, 121, 179 | | SAFECode  8, 10, 11 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  89, 90 | | OWASP ASVS  12.1 | | OWASP AppSensor  - | | CAPEC  75, 133, 203 | | SAFECode  8, 10, 11 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | |
| **Authorization** | **Q** | **Authorization** | **K** |  |  |  |  |
| Christopher can inject a command that the application will run at a higher privilege level | Ryan can influence or alter authorization controls and permissions, and can therefore bypass them | (no card) | (no card) |
| |  | | --- | | OWASP SCP  208 | | OWASP ASVS  4.1, 4.6 | | OWASP AppSensor  - | | CAPEC  17, 30, 69, 234 | | SAFECode  8, 10, 11 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  77, 91 | | OWASP ASVS  4.9, 4.10, 4.11 | | OWASP AppSensor  - | | CAPEC  56, 207, 211 | | SAFECode  8, 10, 11 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Cryptography** | **A** | **Cryptography** |  | **Cryptography** | **2** | **Cryptography** | **3** |
| You have invented a new attack against Cryptography | (no card) | Kyun can access data because it has been obfuscated rather than using an approved cryptographic function | Axel can modify transient or permanent data (stored or in transit), or source code, or updates/patches, or configuration data, because it is not subject to integrity checking |
| *Read more about this topic in OWASP’s free Cheat Sheets on Cryptographic Storage, and Transport Layer Protection* |  | |  | | --- | | OWASP SCP  133, 135 | | OWASP ASVS  - | | OWASP AppSensor  - | | CAPEC  - | | SAFECode  21, 29 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  92, 204, 211, 213 | | OWASP ASVS  12.3, 13.2 | | OWASP AppSensor  SE1, IE4 | | CAPEC  31, 39, 68, 75, 133, 145, 162, 203,438-9,442 | | SAFECode  12, 14 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | |
| **Cryptography** | **4** | **Cryptography** | **5** | **Cryptography** | **6** | **Cryptography** | **7** |
| Paulo can access data in transit that is not encrypted, even though the channel is encrypted | Kyle can bypass cryptographic controls because they do not fail securely (i.e. they default to unprotected) | Romain can read and modify data in transit (e.g. cryptographic secrets, credentials, session identifiers, personal and commercially-sensitive data), in communications within the application, or between the application and users, or between the application and external systems | Gunter can intercept or modify encrypted data in transit because the protocol is poorly deployed, or weakly configured, or certificates are invalid, or certificates are not trusted, or the connection can be degraded to a weaker or un-encrypted communication |
| |  | | --- | | OWASP SCP  - | | OWASP ASVS  - | | OWASP AppSensor  - | | CAPEC  185, 186, 187 | | SAFECode  14, 29, 30 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  103, 145, 147 | | OWASP ASVS  7.2 | | OWASP AppSensor  - | | CAPEC  97 | | SAFECode  21, 29 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  36, 37, 133, 143, 146, 147 | | OWASP ASVS  9.2 | | OWASP AppSensor  - | | CAPEC  31, 57, 102, 158, 384, 466 | | SAFECode  29 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  37, 75, 144, 145, 148, 149 | | OWASP ASVS  10.1, 10.2, 10.3, 10.5, 10.8, 10.9, V11.5 | | OWASP AppSensor  IE4 | | CAPEC  31, 217 | | SAFECode  14, 29, 30 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Cryptography** | **8** | **Cryptography** | **9** | **Cryptography** | **10** | **Cryptography** | **J** |
| Eoin can access stored business data (e.g. passwords, session identifiers, PII, cardholder data) because it is not securely encrypted or securely hashed | Andy can bypass random number generation, random GUID generation, hashing and encryption functions because they have been self-built and/or are weak | Susanna can break the cryptography in use because it is not strong enough for the degree of protection required, or it is not strong enough for the amount of effort the attacker is willing to make | Justin can read credentials for accessing internal or external resources, services and others systems because they are stored in an unencrypted format, or saved in the source code |
| |  | | --- | | OWASP SCP  30, 70, 133, 135, 171 | | OWASP ASVS  2.13, 2.14, 7.4, 8.10, 9.2 | | OWASP AppSensor  - | | CAPEC  31, 37, 55 | | SAFECode  21, 29, 31 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  30, 60, 104, 105 | | OWASP ASVS  7.6, 7.7, 7.8 | | OWASP AppSensor  - | | CAPEC  97 | | SAFECode  14, 21, 29, 32, 33 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  104, 105 | | OWASP ASVS  7.6, 7.7, 7.8 | | OWASP AppSensor  - | | CAPEC  97, 463 | | SAFECode  14, 21, 29, 31, 32, 33 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  35, 171, 172 | | OWASP ASVS  2.14, 12.1 | | OWASP AppSensor  - | | CAPEC  116 | | SAFECode  21, 29 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | |
| **Cryptography** | **Q** | **Cryptography** | **K** |  |  |  |  |
| Randolph can access or predict the master cryptographic secrets | Dan can influence or alter cryptography code/routines (encryption, hashing, digital signatures, random number and GUID generation) and can therefore bypass them | (no card) | (no card) |
| |  | | --- | | OWASP SCP  35, 102 | | OWASP ASVS  7.3 | | OWASP AppSensor  - | | CAPEC  116, 117 | | SAFECode  21, 29 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  31, 101 | | OWASP ASVS  7.1 | | OWASP AppSensor  - | | CAPEC  207, 211 | | SAFECode  14, 21, 29 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Cornucopia** | **A** | **Cornucopia** |  | **Cornucopia** | **2** | **Cornucopia** | **3** |
| You have invented a new attack of any type | (no card) | Lee can bypass application controls because dangerous/risky programming language functions have been used instead of safer alternatives, or there are type conversion errors, or because the application is unreliable when an external resource is unavailable, or there are race conditions, or there are resource initialization or allocation issues, or overflows can occur | Andrew can access source code, or decompile, or otherwise access business logic to understand how the application works and any secrets contained |
| *Read more about application security in OWASP’s free Guides on Requirements, Development, Code Review and Testing, the Cheat Sheet series, and the Open Software Assurance Maturity Model* |  | |  | | --- | | OWASP SCP  194-202, 205-209 | | OWASP ASVS  5.1 | | OWASP AppSensor  - | | CAPEC  25, 26, 29, 96, 123-4, 128-9, 264-5 | | SAFECode  3, 5-7, 9, 22, 25-26, 34 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  134 | | OWASP ASVS  - | | OWASP AppSensor  - | | CAPEC  56, 189, 207, 211 | | SAFECode  - | | OWASP Cornucopia Ecommerce Website Edition v0.3 | |
| **Cornucopia** | **4** | **Cornucopia** | **5** | **Cornucopia** | **6** | **Cornucopia** | **7** |
| Keith can perform an action and it is not possible to attribute it to him | Larry can influence the trust other parties including users have in the application, or abuse that trust elsewhere (e.g. in another application) | Aaron can bypass controls because error/exception handling is missing, or is implemented inconsistently, or is partially implemented, or does not deny access by default (i.e. errors terminate access/execution), or relies on handling by some other service or system | Mwengu's actions cannot be investigated because there is not an adequate accurately time-stamped record of security events, or there is not a full audit trail, or these can be altered or deleted by Mwengu, or there is no centralized logging service |
| |  | | --- | | OWASP SCP  181 | | OWASP ASVS  - | | OWASP AppSensor  - | | CAPEC  - | | SAFECode  - | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  - | | OWASP ASVS  - | | OWASP AppSensor  - | | CAPEC  89, 103, 181, 459 | | SAFECode  - | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  109, 110, 111, 112, 155 | | OWASP ASVS  8.4 | | OWASP AppSensor  - | | CAPEC  54, 98, 164 | | SAFECode  4, 11, 23 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  113-115, 117, 118, 121-130 | | OWASP ASVS  2.12, 4.15, 5.7,7.5,8.3,8.5-6,8.8,8.9,10.4,12.3 | | OWASP AppSensor  - | | CAPEC  93 | | SAFECode  4 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| **Cornucopia** | **8** | **Cornucopia** | **9** | **Cornucopia** | **10** | **Cornucopia** | **J** |
| David can bypass the application to gain access to data because the network and host infrastructure, and supporting services/applications, have not been securely configured, the configuration rechecked periodically and security patches applied, or the data is stored locally, or the data is not physically protected | Michael can bypass the application to gain access to data because administrative tools or administrative interfaces are not secured adequately | Xavier can circumvent the application's controls because code frameworks, libraries and components contain malicious code or vulnerabilities (e.g. in-house, commercial off the shelf, outsourced, open source, externally-located) | Roman can exploit the application because it was compiled using out-of-date tools, or its configuration is not secure by default, or security information was not documented and passed on to operational teams |
| |  | | --- | | OWASP SCP  151, 152, 156, 160, 161, 173-177 | | OWASP ASVS  11.2, 11.3, 11.6 | | OWASP AppSensor  RE1, RE2 | | CAPEC  37, 220, 289, 310, 436 | | SAFECode  - | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  - | | OWASP ASVS  - | | OWASP AppSensor  - | | CAPEC  225, 122 | | SAFECode  - | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  57, 151, 152, 204, 212 | | OWASP ASVS  2.15, 3.13, 4.16, 5.9, 6.10, 7.10, 8.12, 13.1 | | OWASP AppSensor  - | | CAPEC  68, 438, 439, 442 | | SAFECode  15 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  - | | OWASP ASVS  - | | OWASP AppSensor  - | | CAPEC  - | | SAFECode  4 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | |
| **Cornucopia** | **Q** | **Cornucopia** | **K** | **Wild Card** | **Joker** | **Wild Card** | **Joker** |
| Jim can undertake malicious, non-normal, actions without real-time detection and response by the application | Gareth can utilize the application to deny service to some or all of its users | Alice can utilize the application to attack users' systems and data | Bob can influence, alter or affect the application so that it no longer complies with legal, regulatory, contractual or other organizational mandates |
| |  | | --- | | OWASP SCP  - | | OWASP ASVS  - | | OWASP AppSensor  (All) | | CAPEC  (All) | | SAFECode  1, 27 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | |  | | --- | | OWASP SCP  41 | | OWASP ASVS  - | | OWASP AppSensor  - | | CAPEC  2, 25, 119 | | SAFECode  1 | | OWASP Cornucopia Ecommerce Website Edition v0.3 | | *Have you thought about becoming an individual OWASP member? All tools, guidance and local meetings are free for everyone, but individual membership helps support OWASP’s work* | *Examine vulnerabilities and discover how they can be fixed using training applications in the free OWASP Broken Web Applications VM, or using the online challenges in the free Hacking Lab* |

Card trim lines to be printed here…

